Verification and Validation of KBS With Neural Network Components

Wu Wen and **John Callahan**
NASA/WVU Software Research Laboratory
886 Chestnut Ridge Road
Morgantown, WV 26506-6506

Abstract

Artificial Neural Networks(ANN) play an important role in developing robust Knowledge Based Systems(KBS). The ANN based components used in these systems learn to give appropriate predictions through training with correct input-output data patterns. Unlike traditional KBS that depends on a rule database and a production engine, the ANN based system mimics the decisions of an expert without specifically formulating the if-then type of rules. In fact, the ANNs demonstrate their superiority when such if-then type of rules are hard to generate by human expert. Verification of traditional knowledge based system is based on the proof of consistency and completeness of the rule knowledge base and correctness of the production engine. These techniques, however, can not be directly applied to ANN based components. In this position paper, we propose a verification and validation procedure for KBS with ANN based components. The essence of this procedure is to obtain an accurate system specification through incremental modification of the specifications using an ANN rule extraction algorithm. First, the ANN based components are specified using available domain knowledge and implemented based on this specification. Next, past data sets are used to train the ANN components. An rule extraction algorithm is then applied to the trained ANN. Extracted rules are then analyzed and incrementally incorporated into the system specification. Finally, the modified specifications are verified for correctness and the product tested against the correct specifications.

Introduction

This research is motivated by the recent surge in applications of ANNs in real world systems, especially in safety critical systems. Procedures for verification of such system should satisfy criteria that are used to verify conventional systems. This position paper is our first effort in understanding the differences between the conventional and ANN based system. We also attempt to show how this difference can be dealt with in some cases. Currently we are working with a research group here in West Virginia University that uses ANNs to learn how to fly a model airplane. Furthermore, this ANN based system is then used as the autopilot(Napolitano and Kincheloe 1995) to control the airplane. Verification and validation of the autopilot thus designed will be critical for safety purposes.

In designing traditional autopilot systems, different types of controllers and control strategies are applied to different segments of the autopilot operation profile. Autopilot performance within each segment and during transition between segments are vigorously verified and tested for safety. The adaptation of ANN based controller eliminated the need for a segmented design thus simplifies the overall system design.

The ANN based autopilot described above either learns from previously recorded flight data or through on-line learning from an actual pilot. Experimental results have shown that the ANN based approach has the advantages of shorter response time, reduced oscillation and improved robustness. Moreover, it can be proven(Hornik 1991; Cybenko 1989) that feedforward ANN with hidden layer can approximate any input/output relationship given appropriate network topology.

However, a number of questions must be answered before they can be used in such real-world safety critical situations.

- **Completeness of training data**: does the recorded flight data or the on-line data from actually pilot flying contains all possible situations that an airplane will be in?
- **Flight envelope**: what will be the performance of the ANN based autopilot in the vicinity of the boarder region of the flight envelope?
- **Generalization**: neural networks are known to be capable of generalization so patterns that have not appeared in the training data can be dealt with appropriately. However, to what extend this generalization applies depends on a lot of factors, such as the number of units and training methods. How can we verify that which cases were generalized and which were not?

- **Validation**: traditional autopilot system is tested by simulating flight conditions in each segments of the flight envelop. Tests for performance within segment and during transitions are vigorously tested. For ANN based autopilot system, such tests will be difficult to generate. Exhaustive testing will also be difficult if not impossible.

There are two essential problems with verification and validation of any system:

- That the developed system satisfies the specification must be verified;
- That the correctness of the specification must be verified.

To this end, an accurate specification of the system and testable specification-product correspondence must be provided. This is especially true for ANN based system where the specifications are lacking in precision. Furthermore, any changes in the specifications must be identifiable and verifiable in the product and vice versa.

The rest of the paper is organized as follows. In Section 2 we first clarify the term *Verification and Validation of knowledge based systems*. In section 3 we describe how domain knowledge based specifications can be used to implement the fuzzy neural networks. Then we will briefly show how rule extraction can be used to modify the specifications. Section 4 describes a procedure for the verification and validation of ANN based system using fuzzy neural network learning and rule extraction. Finally in section 5 we discuss the limitation of this approach and what our future work will be.

Requirements Specification and Verification

We first briefly describe our example problem. The model airplane to be controlled by the autopilot has the following inputs: altitude, airspeed, acceleration, pitch and roll angle, surface reflection of the rudder and aileron and angle of attack. The outputs include: aileron, rudder and throttle control position. The autopilot system contains the following subsystems: altitude hold subsystem, airspeed hold system, pitch hold system and roll angle control system. Conventional specification of such system contains detailed description of controller behavior at various segments of the flight envelope. For example, the altitude will be divided into taking-off, low altitude, cruise altitude and landing etc. The airspeed will also be divided into taking-off, acceleration, cruising, prepare for landing and landing. Maneuvers and control actions such as turning, ascending and descending are specified in details with respect to each of the segmented flight conditions. It is clear that a vague specification based on the above conventional specification can be used to capture the domain knowledge in specifying the autopilot system. The exact states, boundaries of the segments will not be known at this stage. This level of specification forms a conceptual understanding of the problem at hand.

In software engineering definition, verification is the effort to substantiate that the system correctly implements the system specifications(London 1977; Andrion, Branstad and Chernlavsky 1982). A more detailed definition(Andriole 1986) declares that verification is the determination that the software product is correct, complete, and consistent within itself and its specification. According to IEEE, software verification is the process of determining whether or not the products of a given phase of software development meet all of the requirements established during the previous phase(IEEE 1983).

The general framework for verification of knowledge based system has been described in various works(Andert 1992; Gupta 1991; Laurent and Ayel 1991; Nazareth and Kennedy 1993). It usually consists of a static verification phase and a dynamic verification phase. In static verification the following analysis are usually conducted:

- Consistency checking
  - checking for redundant rules
  - checking for conflicting rules
  - checking for circular logic
  - KB language semantics checking
- Completeness checking
  - checking for dead-end condition
  - checking for unreachable conclusion
  - exhaustive completeness check

Dynamic verification usually involves testing of the decision path for consistency and completeness. The functions of the knowledge base will also be tested against its requirements specification. Sometimes historic data that are generated by human expert are used to generate test cases to further verify the knowledge base.

These verification techniques apply to most of the knowledge based systems that are developed on a knowledge database and a production engine. However, when ANN based components are incorporated into the knowledge based system, the above verification technique can not be used to directly verify such system.

The development of a definitive requirements specification for any real-world software system is difficult and often subject to inaccuracies. ANN based system poses further challenges. The neural networks is supposed to learn and discover the structure and parameters of the problem system at hand through training.

We maintain that the development of a usable requirement specification is a dynamic process that can not be determined completely at the start of the design. Therefore the verification effort must accommodate the dynamic characteristics and facilitate the formulation

and revision of the requirement specification. This is especially true for real world systems that are hard to specify in definitive terms and must evolve along the development cycle. The ANN based autopilot system is such an example. The initial requirement specification of such system can be as simple as "This ANN based autopilot system should behave as closely to the human pilot it learns from as possible".

It can be argued that the purpose of using such learning based approach is to eliminate the need for an detailed specification. However, from an engineering point of view, if such specifications can not be verified or tested, alternative specifications must be found to this end. We therefore require that the specifications describe behavior of the system with respect to its functional states. These functional states must be complete but can be vague and at a high level in the beginning. As much as possible domain related knowledge must be incorporated at this stage. The rest of this paper describes a tentative approach to maintain a system representation which is conducive to the specification and which allows such specifications be detailed, modified and verified.

Mapping Requirement Specifications To Fuzzy Neural Networks

Fuzzy logic was invented to allow people to make generalizations about system behavior in linguistic terms and manipulate them with approximate reasoning(Zadeh 1965). In fact, many natural language requirement specifications can be regarded as fuzzy. It must be pointed out that the fuzzy description of the system behavior can be consistent and complete without being specific. For many complex real-world problem, the detailed specifications may be difficult to obtain at the time of the design. That is why traditional fuzzy systems are designed and implemented through trial-and-error.

On the other hand fuzzy neural networks(Cox 1992; Kosko 1992; Khan 1993) have gained considerable acceptance in recent years due to the fact that it combines the domain knowledge based approach of fuzzy logic and self organizing and self adapting property of neural networks. One of the main objectives of using fuzzy neural nets is to use the neural nets to solve a problem and at the same time explain how it is done. In (Kong and Kosko 1992) an fuzzy neural nets approach is used to learn the control strategies of the backing-up of a truck-and-trailer and to extract fuzzy rules of these strategies. The authors claimed that the fuzzy neural networks not only can learn how to back up the truck, but also can tell you what it has learned. For a detailed discussion on fuzzy neural networks please refer to (Kosko 1992).

It must be pointed out that these is nothing fuzzy about the fuzzy logic rules in terms of its mapping of the input-output relationships. Once the fuzzy sets and its associated membership functions are determined, the fuzzy logic system describes a unique mapping between the input and output state spaces. It can be proven(Kosko 1992) that with appropriate choice of the fuzzy sets and their membership functions, fuzzy logic based system can approximate any bounded and continuous function. Conventional fuzzy logic system requires human fuzzy engineers to determine the fuzzy sets and the fuzzy membership function. These rules and functions are then modified based on trial-and-error. By combining neural networks with fuzzy logic, fuzzy logic based system can be automatically configured through learning.

For our example of specifying the autopilot, the highly non-linearity of the flight system means that a definitive description of the system behavior will be difficult to obtain. It is, however, possible to describe its behavior in a vague term based on our domain knowledge of the flight control system. Input variables such as altitude, airspeed, angle of attack etc can be modeled adequately by fuzzy sets. For example, altitude can be modeled as a fuzzy set containing taking-off, low, cruise, landing etc; airspeed can be taking-off, ascending, cruise and landing. The boundary and membership function that will uniquely describe the input state space will be determined by the ANN based learning.

Specifications such as safety constraints and procedures are harder to map onto the fuzzy set since they are dependent on the dynamic behavior of the system. For example, in traditional autopilot design, requirement specifications may include sentences such as "Under no circumstance should the throttle be advanced if the airplane is ascending with high angle of attack and low airspeed". However, certain constrains can be readily implemented as fixed weight connections on the units of ANN. In any case, such a high level specification allows safety constrains such as those described above to be incorporated into the system representation. Verification can then be conducted once the boundaries and functions of the fuzzy sets are found.

The fuzzy neural network system development paradigm is very important to the verification and validation of complex, real-world system development.

- it allows the full integration of expert domain knowledge into specification;
- it accommodates gradual modifications on both specification and implementation through fuzzy neural nets rule extraction.

The close coupling of specification and implementation allows effective verification and validation to be conducted.

Rule Extraction in Neural Networks

Recent development in neural networks research has shown that it is possible to extract simple input/output relationships/rules by pruning the number of connections and discretizing hidden-unit activation values

through clustering(Setiono and Liu 1996; Towell and Shavlik 1993). In (Setiono and Liu 1996) an algorithm called NeuoRule is developed to extract simple diagnostic decision rules from human expert diagnosis database. The NeuoRule algorithm is based on a standard three-layer feed-forward network. The rule extraction consists of four phases:

- build a weight-decay back-propagation network so that weights reflect the importance of the networks' connection;
- prune the network to remove irrelevant connections and units while maintaining the network's accuracy;
- discretize the hidden-unit activation values by clustering;
- extracts rules from the network with discretized hidden-unit activation values.

The neural network rule extraction described above has significant role in verification and validation of knowledge based systems that uses neural networks. Through rule extraction, domain knowledge based specifications can be further verified against the actual behavior of the system. New rules discovered and extracted by the NeuroRule algorithm will be used to modify the specifications.

Verification of the Neural Network Based System

Based on the domain knowledge based fuzzy specification and the rule extraction algorithm, we propose an incremental procedure for the verification and validation of ANN based systems. This is illustrated in Figure 1.

Fuzzy specification

The domain knowledge that is used in building conventional system can be exploited. Based on the conventional specification, fuzzy sets can be derived to describe the high level behavior of the system. This fuzzy set specification can be checked for completeness and consistency using formal verification tools such as PVS(Crow et al. 1995).

Multiple Neural Nets

The ANN based autopilot described in (Napolitano and Kincheloe 1995) is implemented on a distributed neural networks. The modules for altitude control, speed control and turning etc are implemented on different neural network modules that are trained separately. Conventional autopilot specifications include detailed description of behaviors during mode transition to ensure safety. Fuzzy specifications allow such high level transition rules to be incorporated. These are examples of such rules:

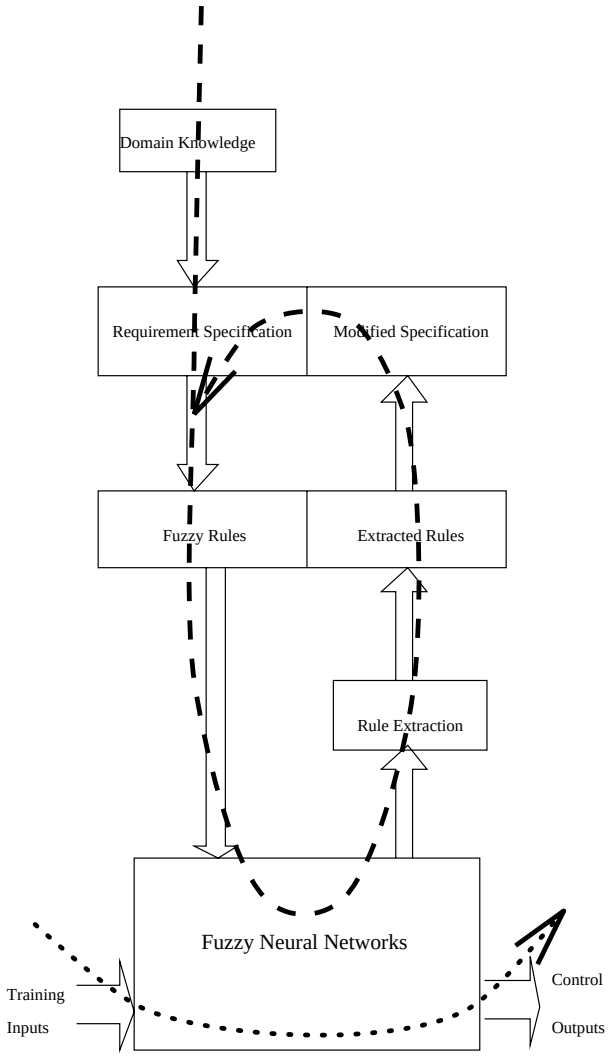

Figure 1: Incremental Modification of Specification and Specification-product Correspondence

- When angle of attack is too high and airspeed is too low, transit from ascending mode to horizontal mode;
- The transition period from ascending mode to horizontal mode must not exceed a fraction of a second.

The correctness and safety properties such as those described above can be verified using tools such as the *SPINTOOL*(Holzmann 1994) once the time delay can be determined from the learned system.

Fuzzy Neural Network Learning

The membership functions of the fuzzy sets can then be learned using data generated from the trained ANNs. For this procedure the NeuroFuz tool(Khan 1993) can be used to determine the membership functions and prune the redundant fuzzy rules.

Rule Extraction

In order to verify that the fuzzy specifications do contain all the necessary input-output relationships embedded in the real system, rule extraction algorithm based on NeuroRule(Setiono and Liu 1996) can be used. The extracted rules are then compared with the learned fuzzy rules. Conflicts and inconsistencies must be resolved by human expert at this stage and fuzzy specification modified.

Verification

The above procedure iterates until the fuzzy rules and the extracted rules are consistent. The modified fuzzy specification is then further checked for consistency and completeness. Mode transition and safety properties are also checked for the modified specification. Finally we arrive at an ANN based system that is consistent with its specification.

Discussion and Future Work

This position paper describes an incremental procedure for verifying knowledge based system with ANN based components. First, domain knowledge is converted into fuzzy specifications that describe the high level behavior of the system. Next, fuzzy neural network tools are used to obtain accurate membership functions and simplify the fuzzy rules. Furthermore, rules are extracted from the ANNs using NeuroRule and compared with the fuzzy rules. Fuzzy specifications are then modified and the results compared again. Finally the fuzzy specification is checked for consistency, correctness and safety.

The appropriateness of this approach will depend on the applicability of the fuzzy neural networks and the rule extraction algorithm. Domain knowledge also plays an important role in giving a more accurate specification to begin with. Unfortunately at this stage, we are not able to report on the details of the application of this approach due to the short time we have embarked on this project. Currently we are training the ANNs using a simple conventional rule based autopilot. We plan to use the rule extraction algorithm to extract rules from the above learned ANNs and compare them with the simple rules that was used to train the ANNs. If these simple rules that were used to drive the auto-pilot can be extracted, we can then proceed in applying the verification procedure described in this paper for the ANN based auto-pilot.

References

W. Adrion, M. Branstad, and J. Chernlavsky. Validation, verification and testing of computer software. *ACM Computing Surveys*, 14:159–192, 1982.

S. Andriole. *Software Validation, Verification, Testing and Documentation*. Petrocelli Books, 1986.

E. P. Anert. Integrated knowledge-based system design and validation for solving problems in uncertain environments. *Int. J. of Man-Machine Studies*, 36:357–373, 1992.

E. Cox. Integrating fuzzy logic into neural nets. *AI Expert*, pages 43–47, 1992.

J. Crow, S Owre, J. Rushby, N. Shankar, and M. Srivas. A tutorial introduction to pvs. In *Workshop on Industrail-Strenth Formal Specification Techniques*, Boca Raton, Florida, Apri 1995.

G. Cybenko. Approximation by superposition of signoidal functions. *Mathematics of Control Signals and Systems*, 2(4):303, 1989.

U. Gupta. Validating and verifying knowledge-based systems. In *IEEE Computer Society Press*, Los Alamitos, California, 1991.

G. J. Holzmann. Basic spin manual. Technical report, 1994.

K. Hornik. Approximation capabilities of multilayer feedforward neural networks. *Neural Networks*, 1:251–257, 1991.

IEEE. Standard glossary for software engineering terminology. *IEEE Standard*, page 729, 1983.

E. Khan. Neufuz: An intelligent combination of fuzzy logic with neural nets. In *IJCNN93*, Nagoya, Japan, Oct 1993.

S. Kong and B. Kosko. Adaptive fuzzy systems for backing up a truck-and-trailer. *IEEE Transactions on Neural Networks*, 3(2):211–223, 1992.

B. Kosko. *Neural Networks and Fuzzy Logic*. Pentice Hall, 1992.

J. P. Laurent and M. Ayel. *Verification, Validation and Test of KBS*. John Wiley, 1991.

R. L. London. *Current Trends in Programming Methodology: Volume 2 Program Validation*, chapter Perspectives on Program Verification. Prentice Hall, 1977.

M. R. Napolitano and M. Kincheloe. On-line learning nueral network controllers for autopilot systems. In *95' AIAA Guidance Navigation and Control Conference*, Baltimore, Md, August 1995.

D. L. Nazareth and M. H. Kennedy. Knowledge-based system verification, validation and testing: The evolution of a discipline. *Int. J. of Expert Systems*, 6(2):143–162, 1993.

R. Setiono and H. Liu. Symbolic representation of neural networks. *IEEE Computer Magzine*, pages 71–77, 1996.

G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, 1993.

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.